Building for Voice with Siri Shortcuts

Session 214

Amit Jain, Siri
Ayaka Nonaka, Siri
Adding shortcuts to Siri

Custom responses

Best practices for shortcuts

Bringing shortcuts into your app
Adding shortcuts to Siri

Custom responses

Best practices for shortcuts

Bringing shortcuts into your app
Adding shortcuts to Siri

Custom responses

Best practices for shortcuts

Bringing shortcuts into your app
Adding shortcuts to Siri
Custom responses
Best practices for shortcuts
Bringing shortcuts into your app
Adding Shortcuts to Siri
Soup Menu

- New England Clam Chowder
- Tomato Soup
Add to Siri
Order 1 tomato soup with red pepper
Soup Chef

You can say something like...
“Soup time”
“Soup time”
Order 1 tomato soup with red pepper

“Soup time”
“Soup time”
Order 1 tomato soup with red pepper

“Important emails”
VIP

“Caffeinate me”
Search for Coffee

“Lunch time”
Order soup of the day
Soup time

Ready to order with Soup Chef?

Order 1 tomato soup with red pepper

Cancel   Order
Soup time

Ready to order with Soup Chef?

Order 1 tomato soup with red pepper

OK. Ordering with Soup Chef.
Custom Responses
Types of custom responses

Confirmation
Success
Informational
Error
Bean says: "Your coffee will be $4.95." Ready to order?
Types of custom responses
Success

Bean says: "You can pick up your latte in 10 minutes."
Types of custom responses
Information

Commute says: “35 Eureka will be at the Castro and 17th stop in 4 minutes.”
Something went wrong. Bean says, “The Cupertino store is currently closed.” You can continue in the app.
Adopting Custom Responses
Adopting Custom Responses

Defining a custom response

Define a custom intent
Adopting Custom Responses
Defining a custom response

Define a custom intent
Adopting Custom Responses

Defining a custom response

Define a custom intent

Select a category for your intent
Adopting Custom Responses
Defining a custom response

Define a custom intent
Select a category for your intent
Adopting Custom Responses
Defining a custom response

Define a custom intent
Select a category for your intent
Declare response properties
### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Array</th>
</tr>
</thead>
</table>

### Response Templates

<table>
<thead>
<tr>
<th>Code</th>
<th>Success</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>failure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>success</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

+---+---+---+---+---+
|   |   |   |   |   |

+---+---+---+---+---+
<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Property</td>
<td>Type</td>
<td>Array</td>
</tr>
<tr>
<td>----------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>waitTime</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>soup</td>
<td>Custom</td>
<td></td>
</tr>
</tbody>
</table>
Adopting Custom Responses
Defining a custom response

Define a custom intent
Select a category for your intent
Declare response properties
Adopting Custom Responses
Defining a custom response

Define a custom intent
Select a category for your intent
Declare response properties
Provide a response template string
### CUSTOM INTENTS

<table>
<thead>
<tr>
<th>OrderSoup</th>
<th>Response</th>
</tr>
</thead>
</table>

#### Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>soup</td>
<td>Custom</td>
<td></td>
</tr>
</tbody>
</table>

#### Response Templates

<table>
<thead>
<tr>
<th>Code</th>
<th>Success</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>failure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>success</td>
<td>☑️</td>
<td>Your soup will be ready in <code>waitTime</code></td>
</tr>
</tbody>
</table>

+ | -
## Response Templates

<table>
<thead>
<tr>
<th>Code</th>
<th>Success</th>
<th>Template</th>
</tr>
</thead>
<tbody>
<tr>
<td>failure</td>
<td></td>
<td></td>
</tr>
<tr>
<td>success</td>
<td>✔️</td>
<td>Your <strong>soup</strong> will be ready in <strong>waitTime</strong>.</td>
</tr>
<tr>
<td>failureOutOfStock</td>
<td>✔️</td>
<td>Sorry, <strong>soup</strong> is out of stock.</td>
</tr>
</tbody>
</table>
// Custom response object
public class OrderSoupIntentResponse : INIntentResponse {
    public var waitTime: String? { get set }
    public var soup: INObject? { get set }

    public convenience init(code: OrderSoupIntentResponseCode, userActivity: NSUserActivity?)

    public static func success(soup: INObject, waitTime: String) -> OrderSoupIntentResponse

    public static func failureOutOfStock(soup: INObject) -> OrderSoupIntentResponse
}
public class OrderSoupIntentResponse : INIntentResponse {
    public var waitTime: String? { get set }
    public var soup: INObject? { get set }

    public convenience init(code: OrderSoupIntentResponseCode, userActivity: NSUserActivity?)

    public static func success(soup: INObject, waitTime: String) -> OrderSoupIntentResponse

    public static func failureOutOfStock(soup: INObject) -> OrderSoupIntentResponse
}
// Custom response object

public class OrderSoupIntentResponse : INIntentResponse {
    public var waitTime: String? { get set }
    public var soup: INObject? { get set }

    public convenience init(code: OrderSoupIntentResponseCode, userActivity: NSUserActivity?)

    public static func success(soup: INObject, waitTime: String) -> OrderSoupIntentResponse

    public static func failureOutOfStock(soup: INObject) -> OrderSoupIntentResponse
}
Adopting Custom Responses

Defining a custom response

Define a custom intent

Select a category for your intent

Declare response properties

Provide a response template string
Adopting Custom Responses
Defining a custom response

Define a custom intent

Select a category for your intent

Declare response properties

Provide a response template string

Provide custom responses in your intent handler
/ Confirm that the intent can be run

class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}

// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}
// Confirm that the intent can be run

class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}
// Confirm that the intent can be run

class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}

let soupMenuManager = SoupMenuManager()
let menuItem = soupMenuManager.findItem(identifier: identifier)

if menuItem.isAvailable == false {
    completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
    return
}

completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))

}
// Confirm that the intent can be run

class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}
// Confirm that the intent can be run

class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}

// Confirm that the intent can be run
class OrderSoupIntentHandler: NSObject, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion:(OrderSoupIntentResponse) -> Void) {
        guard let identifier = intent.soup?.identifier else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let soupMenuManager = SoupMenuManager()
        let menuItem = soupMenuManager.findItem(identifier: identifier)

        if menuItem.isAvailable == false {
            completion(OrderSoupIntentResponse.failureOutOfStock(soup: soup))
            return
        }

        completion(OrderSoupIntentResponse(code: .ready, userActivity: nil))
    }
}
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)

        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
// Handle the intent

class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)

        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
// Handle the intent

class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)
        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard let soup = intent.soup,
             let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)
        completion(OrderSoupIntentResponse(code: .success, userActivity: nil))
    }
}
class IntentHandler: INExtension, OrderSoupIntentHandling {

    func confirm(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        ...
    }

    func handle(intent: OrderSoupIntent, completion: (OrderSoupIntentResponse) -> Void) {
        guard 
            let soup = intent.soup,
            let order = Order(from: intent)
        else {
            completion(OrderSoupIntentResponse(code: .failure, userActivity: nil))
            return
        }

        let orderManager = SoupOrderDataManager()
        orderManager.placeOrder(order: order)
        completion(OrderSoupIntentResponse.success(soup: soup, waitTime: order.waitTime))
    }

    let orderManager = SoupOrderDataManager()
    orderManager.placeOrder(order: order)
}
Soup time

Ready to order with Soup Chef?

Order 1 tomato soup with red pepper

OK. Placing an order with Soup Chef.
Soup time

Ready to order with Soup Chef?

Order 1 tomato soup with red pepper

OK. Placing an order with Soup Chef.

OK. Ordering. Soup Chef says:
"Your tomato soup will be ready in 10 minutes."
Sorry, there was a problem with the app.
Sorry, there was a problem with the app.

Something went wrong. Soup Chef says, "Sorry, we're all out of tomato soup." You can continue in the app.

Open Soup Chef
Adopting Custom Responses
Custom UI Extension

Provide a custom UI for your shortcut

Shows in Siri, on Lock Screen, and in Search
Adopting Custom Responses
Custom UI Extension

Provide a custom UI for your shortcut

Shows in Siri, on Lock Screen, and in Search
Soup time

Ready to order with Soup Chef?

Order 1 tomato soup with red pepper

Cancel  Order
Soup time

Ready to order with Soup Chef?

Order 1 tomato soup with red pepper

OK. Placing an order with Soup Chef.
Soup time
Ready to order with Soup Chef?

SOUP CHEF
Order 1 tomato soup with red pepper

OK. Placing an order with Soup Chef.

Soup time
Ready to order with Soup Chef?

SOUP CHEF
Tomato Soup
Order Confirmed
Wait time 10 Minutes
Total $2.95

OK. Placing an order. Soup Chef says: "Your tomato soup will be ready in 10 minutes."
Demo
Adopting Custom Responses

Define a custom intent
Select a category for your intent
Declare response properties
Provide a response template string
Provide the response in your intent handler
Siri Shortcuts Best Practices

Ayaka Nonaka, Siri
NSUserActivity
NSUserActivity

Intents
Customizing Donations

Order clam chowder
To One Apple Park Way
Customizing Donations

Title
Subtitle
Image
Suggested invocation phrase

Order clam chowder
To One Apple Park Way
Choosing a Good Title and Subtitle
Choosing a Good Title and Subtitle

The title should represent what happens when you run the shortcut
Choosing a Good Title and Subtitle

The title should represent what happens when you run the shortcut.

The subtitle should provide supplementary information if needed.
Basic Guidelines for Title and Subtitle
Basic Guidelines for Title and Subtitle

Use sentence case
Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise
Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise

Include critical information
Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise

Include critical information

Include a verb for Intents
Basic Guidelines for Title and Subtitle

Use sentence case

Keep the title concise

Include critical information

Include a verb for Intents

Localize
Things to Avoid in Title and Subtitle
Things to Avoid in Title andSubtitle

The name of your app
Things to Avoid in Title and Subtitle

The name of your app

Duplicate information in the title and subtitle
Things to Avoid in Title and Subtitle

The name of your app

Duplicate information in the title and subtitle

“Quotation marks” except when referring to strings that will be used verbatim in the shortcut
A “clam chowder” from Soup Chef

The best way to get soup

Soup Chef
A "clam chowder" from Soup Chef

The best way to get soup

Soup Chef
A "clam chowder"

The best way to get soup

Soup Chef
A “clam chowder”
The best way to get soup

Soup Chef
Order "clam chowder"

The best way to get soup

Soup Chef
Order “clam chowder”
The best way to get soup
Soup Chef
Order clam chowder

The best way to get soup

Soup Chef
Order clam chowder

The best way to get soup

Soup Chef
Order clam chowder
To One Apple Park Way
Soup Chef
Order clam chowder
To One Apple Park Way
Soup Chef
<table>
<thead>
<tr>
<th>Shortcut</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order clam chowder</td>
<td>One Apple Park Way</td>
</tr>
<tr>
<td>Order tomato soup</td>
<td>123 Soup Street</td>
</tr>
</tbody>
</table>
SOUP CHEF

- Order clam chowder
  To One Apple Park Way

- Order tomato soup
  To 123 Soup Street
let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
// Setting an image on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

viewController.userActivity = userActivity
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
    // Handle error
}
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
    // Handle error
}
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
    // Handle error
}
// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
    // Handle error
}

// Setting an image on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
  // Handle error
}
Add to Siri
Order clam chowder
To One Apple Park Way
Soup Chef

After you record your personalized phrase, Siri can use it to tell “Soup Chef” to run this shortcut.
You can say something like...

“Chowder time”
let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons"  // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

userActivity.suggestedInvocationPhrase = "Chowder time"

viewController.userActivity = userActivity
// Suggesting an invocation phrase on an NSUserActivity

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")
userActivity.isEligibleForSearch = true
userActivity.isEligibleForPrediction = true
userActivity.title = "View clam chowder"

let attributes = CSSearchableItemAttributeSet(itemContentType: kUTTypeItem as String)
attributes.contentDescription = "With croutons" // Subtitle

let image = UIImage(named: "Chowder")!
attributes.thumbnailData = image.pngData()

userActivity.contentAttributeSet = attributes

userActivity.suggestedInvocationPhrase = "Chowder time"

viewController.userActivity = userActivity
// Suggesting an invocation phrase on an INIntent

```swift
import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

intent.suggestedInvocationPhrase = "Chowder time"

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in
    // Handle error
}
```
// Suggesting an invocation phrase on an INIntent

import Intents

let intent = PlaceOrderIntent()
let soup = order.soup
intent.soup = INObject(identifier: soup.identifier.uuidString, displayString: soup.name)

intent.setImage(INImage(named: "Chowder"), forParameterNamed: "soup")
intent.setImage(INImage(named: "Office"), forParameterNamed: "deliveryLocation")

intent.suggestedInvocationPhrase = "Chowder time"

let interaction = INInteraction(intent: intent, response: nil)
interaction.donate { error in // Handle error}

Choosing a Good Suggested Invocation Phrase
Choosing a Good Suggested Invocation Phrase

Short and memorable
Choosing a Good Suggested Invocation Phrase

Short and memorable

Don’t include “Hey Siri”
“Hey Siri, please place an order, thank you”
Hey Siri, please place an order, thank you
Order a clam chowder to my office
“Order a clam chowder to my office”
Chowder time
“Chowder time”
「チャウダー食べたい」
<table>
<thead>
<tr>
<th>Session Title</th>
<th>Room</th>
<th>Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Localization Workflows in Xcode 10</td>
<td>Hall 3</td>
<td>WWDC 2018</td>
</tr>
<tr>
<td>Internationalization Lab</td>
<td>Technology Lab 9</td>
<td>Wednesday 11:00AM</td>
</tr>
</tbody>
</table>
Localization

Localize both your code and the strings in your intent definition files

New Localization Workflows in Xcode 10
Hall 3
WWDC 2018

Internationalization Lab
Technology Lab 9
Wednesday 11:00AM
Localization

Localize both your code and the strings in your intent definition files

Pluralization

<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Localization Workflows in Xcode 10</td>
<td>Hall 3</td>
<td>WWDC 2018</td>
</tr>
<tr>
<td>Internationalization Lab</td>
<td>Technology Lab 9</td>
<td>Wednesday 11:00AM</td>
</tr>
</tbody>
</table>
“Chowder time”
Order clam chowder
To One Apple Park Way

Order to One Apple Park Way
Start a new order

Order tomato soup
To 123 Soup Street

Order to 123 Soup Street
Start a new order

Order soup of the day
// How to suggest a shortcut for something a user hasn’t done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]

INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
// How to suggest a shortcut for something a user hasn’t done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]

INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
// How to suggest a shortcut for something a user hasn’t done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]

INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
// How to suggest a shortcut for something a user hasn’t done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]

INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
// How to suggest a shortcut for something a user hasn’t done yet

import Intents

let suggestions = [
    INShortcut(intent: orderFavoriteSoupIntent),
    INShortcut(userActivity: orderFavoriteBeverageUserActivity),
]

INVoiceShortcutCenter.shared.setShortcutSuggestions(suggestions)
Order soup of the day

It's soup time!

Send Message
Recipients: Amit Jain,

Soup Chef

Order clam chowder
To One Apple Park Way

Order to One Apple Park Way
Start a new order

Order tomato soup
To 123 Soup Street

Order to 123 Soup Street
Start a new order
Bringing Shortcuts Into Your App
Thanks for your order!

Your clam chowder will be delivered to One Apple Park Way soon.
Add to Siri
Order clam chowder
To One Apple Park Way
Soup Chef

You can say something like...
“Chowder time”

After you record your personalized phrase, Siri can use it to tell “Soup Chef” to run this shortcut.
/ Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorn.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorn.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an NSUserActivity

import IntentsUI

let userActivity = NSUserActivity(activityType: "com.unicorny.SoupChef.viewOrder")

// Configure your NSUserActivity

let shortcut = INShortcut(userActivity: userActivity)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
/ Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
// Adding a shortcut from an INIntent

import IntentsUI

let intent = PlaceOrderIntent()

// Configure your INIntent

let shortcut = INShortcut(intent: intent)

let viewController = INUIAddVoiceShortcutViewController(shortcut: shortcut)
viewController.delegate = self
present(viewController, animated: true)
Add to Siri
Order clam chowder
To One Apple Park Way
Soup Chef

You can say something like...
“Chowder time”

After you record your personalized phrase, Siri can use it to tell “Soup Chef” to run this shortcut.
Order clam chowder
To One Apple Park Way
Soup Chef

“Chowder time”
Edit

Delete Shortcut
Soup Menu

- Clam chowder
  - "Chowder time"
- Tomato soup
  - "Tomato tomato"
Summary
Summary

Custom responses
Summary

Custom responses

Create rich intents UI
Summary

Custom responses

Create rich intents UI

Details matter
Summary

Custom responses

Create rich intents UI

Details matter

Bring shortcuts into your app
More Information


<table>
<thead>
<tr>
<th>Event</th>
<th>Location</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Siri Shortcuts on the Siri Watch Face</td>
<td>Hall 3</td>
<td>Wednesday 11:00AM</td>
</tr>
<tr>
<td>Shortcuts Lab</td>
<td>Technology Lab 7</td>
<td>Wednesday 12:00PM</td>
</tr>
<tr>
<td>Shortcuts Lab</td>
<td>Technology Lab 7</td>
<td>Thursday 11:00 AM</td>
</tr>
</tbody>
</table>